

PTO 04-3950

Japanese Patent

Document No. H03-282733

RULE GENERATION DEVICE FOR UTILIZING A SOFTWARE COMPONENT

[Sofutowea Buhin Riyo Ruru Seisei Sochi]

Mihoko Kishi and Koichi Kachi

UNITED STATES PATENT AND TRADEMARK OFFICE

Washington, D.C.

June 2004

Translated by: Schreiber Translations, Inc.

<u>Country</u>	:	Japan
<u>Document No.</u>	:	H03-282733
<u>Document Type</u>	:	Kokai
<u>Language</u>	:	Japanese
<u>Inventor</u>	:	Mihoko Kishi and Koichi Kachi
<u>Applicant</u>	:	Toshiba Corp.
<u>IPC</u>	:	G 06 F 9/06
<u>Application Date</u>	:	December 12, 1991
<u>Publication Date</u>	:	March 30, 1990
<u>Foreign Language Title</u>	:	Sofutowea Buhin Riyo Ruru Seisei Sochi
<u>English Title</u>	:	RULE GENERATION DEVICE FOR UTILIZING A SOFTWARE COMPONENT An inter-package

Specification

1. Title of the invention

Rule generation device for utilizing a software component.

2. Patent Claim

A rule generation device for utilizing a software component characterized by the possession of

A component packaging mechanism designed to collect & package software components provided, in relation to the respective targets handled in a given field, for operating said targets,

An inter-component utility restriction scripting mechanism designed to script restrictions on the joint utilities of multiple software components within each component package generated by said component packaging mechanism,

An inter-package relationship definition mechanism designed to define the relationships among the respective component packages generated by the aforementioned component packaging mechanism, and

An inter-component utility restriction fusion mechanism designed to extract, from the script texts of the restrictions on the joint utilities of multiple software components within each component package scripted by the aforementioned inter-component utility restriction scripting mechanism, script portions on the software components actually utilized for program formulation and

¹ Numbers in the margin indicate pagination in the foreign text.

to generate, by fusing these script portions based on the relationships among the respective component packages defined by the aforementioned inter-package relationship definition mechanism, a restriction script text on the joint utilities of multiple software components actually utilized for the aforementioned program formulation.

3. Detailed explanation of the invention

(Objective of the invention)

(Industrial application fields)

The present invention concerns a rule generation device for generating a rule for synthesizing a program formulated by coupling software components, which is used for verifying a program formulated by coupling software components.

(Prior art)

A program formulated in the prior art by coupling software components is verified based on any of the following methods:

<1>: After a program has been formulated by synthesizing components, it is verified in compliance with test specifications according to procedures comparable to those for verifying a program formulated by an ordinary method without recourse to component synthesis;

<2>: After a program has been formulated by synthesizing components, whether or not restrictive conditions on component utility are being satisfied is checked; /2...

<3>: Restrictive conditions among the respective components are preliminarily defined, and whether or not such restrictive conditions are being satisfied by a given component is checked on an occasion for utilizing said component.

The respective methods listed above, however, are plagued with the following problems.

Despite the fact that a program has been formulated by means of component synthesis according to the method of <1>, the advantage of this history cannot be inherited by the test. In other words, despite the guaranteed reliability of each component, it is necessary to formulate, in a redundant manner, test specifications comparable to those of a program formulated without recourse to component synthesis.

The method of <2> is a test method that exploits the advantage of the component synthesis, for only the combination of components with guaranteed qualities needs to be checked, although it is difficult to automate the test in that it is cumbersome to formally script the entire relationships among large numbers of components.

The method of <3> is advantageous in the sense that a program with a high reliability can be formulated since the component synthesis is perpetually checked, although it is difficult to automate the test, as in the method of <2>.

(Problems to be solved by the invention)

Thus, it has been difficult to automate operations of the prior art for verifying programs formulated by means of component synthesis due to the cumbersomeness of formally scripting restrictions on the utilities among the entire components.

The objective of the present invention, which has been conceived for solving these problems, is to provide a rule generation device for utilizing a software component capable of automating program verifying operations.

(Constitution of the invention)

(Mechanism for solving the problems)

(Functions)

The rule generation device of the present invention for utilizing a software component possesses, for the purpose of achieving the aforementioned objective, a component packaging mechanism designed to collect & package software components provided, in relation to the respective targets handled in a given field, for operating said targets, an inter-component utility restriction scripting mechanism designed to script restrictions on the joint utilities of multiple software components within each component package generated by said component packaging mechanism, an inter-package relationship definition mechanism designed to define the relationships among the respective component packages generated by the aforementioned component packaging mechanism, and

an inter-component utility restriction fusion mechanism designed to extract, from the script texts of the restrictions on the joint utilities of multiple software components within each component package scripted by the aforementioned inter-component utility restriction scripting mechanism, script portions on the software components actually utilized for program formulation and to generate, by fusing these script portions based on the relationships among the respective component packages defined by the aforementioned inter-package relationship definition mechanism, a restriction script text on the joint utilities of multiple software components actually utilized for the aforementioned program formulation.

(Functions)

As far as the rule generation device of the present invention is concerned, for utilizing a software component is concerned, software components for operating targets handled in a given field are collected & packaged by the component packaging mechanism in relation to the respective targets, whereas the relationships among the respective component packages are defined by the inter-package relationship definition mechanism, whereas restrictions on the joint utilities of multiple software components within each component package are scripted by the inter-component utility restriction scripting mechanism. The inter-component utility restriction fusion mechanism, furthermore, extracts, from the script texts of the restrictions on the joint utilities of

multiple software components within each component package scripted by the inter-component utility restriction scripting mechanism, script portions on the software components actually utilized for program formulation and generates, by fusing these script portions based on the relationships among the respective component packages defined by the inter-package relationship definition mechanism, a restriction script text on the joint utilities of multiple software components actually utilized for the aforementioned program formulation.

According to the present invention, therefore, it becomes possible to automatically formulate, by using a formal language, the restriction script texts on the joint utilities of multiple software components utilized for the aforementioned program formulation and, as a result, to automate the program verifying operation.

(Application examples)

In the following, an application example of the present invention will be explained with reference to figures.

Figure 1 is a block diagram provided for explaining the constitution of an application example of the rule generation device of the present invention for utilizing a software component.

/3

In the same figure, (1) is a component packaging unit which collects & packages software components (hereafter referred to simply as "components") used for operating data that define

targets handled in a given field (i.e., application program) in relation to the respective targets.

(2) is an inter-component utility restriction scripting unit which scripts, by using a formal language, restrictions on the joint utilities of multiple components within each component package generated by the component packaging unit (1).

(3) is an inter-package relationship definition unit which defines the relationships among the respective component packages generated by the component packaging unit (1) [e.g., parent-offspring (is-a) relationship, etc.]. It should be noted that the relationships among targets hereby serve as the relationships among targets as they are.

(4) is an inter-component utility restriction fusion unit which extracts, from the script texts of the restrictions on the joint utilities of multiple software components within each component package scripted by the inter-component utility restriction scripting unit (2), script portions on the software components actually utilized for program formulation and generates, by fusing these script portions based on the relationships among the respective component packages defined by the inter-package relationship definition unit (3), a restriction script text on the joint utilities of multiple software components actually utilized for the aforementioned program formulation.

Next, the actions of the rule generation device for utilizing a software component characterized by the foregoing constitution

will be explained with reference mainly to Figure 2 as well as to Figures 3 through 7.

First, the component packaging unit (1) extracts targets handled in a given field, whereas the relationships among the respective component packages are defined by the inter-package relationship definition unit (3) (step a). Figure 3 shows examples of such relationships among the respective packages (relationships of targets handled by a window system).

Subsequently, the component packaging unit (1) formulates a component package by collecting components for operating each target (step b). Figure 4 shows a component package for operating an "I/O sub-window," which represents one of such targets.

Next, the inter-component utility restriction scripting unit (2) scripts, by using a formal language, restrictions on the joint utilities of multiple components within each package (step c) and then stores them within a database (step d).

The corresponding example is shown in Figures 5 and 6.

Figure 5 shows a restriction script text on the joint utilities of multiple components within the "main window" package. The following are expressed by this text:

"In order to utilize a component within this package, it is necessary to use 'window-create,' namely a component for securing a main window, in the first place";

"In a case where a component within this package is used, it is necessary to use 'window-destroy,' namely a component for releasing the main window region, at the end."

Figure 6, furthermore, shows a restriction script text on the joint utilities of multiple components within the "I/O sub-window" package.

The following are expressed by this text:

"In order to utilize a component within this package, it is necessary to use 'io-open,' namely a component for opening the I/O sub-window, in the first place";

"In a case where a component within this package is used, it is necessary to use 'io-close,' namely a component for closing the I/O sub-window, at the end."

Subsequently, the inter-component utility restriction fusion unit (4) proceeds, upon the reception of a list of utilized components from the program formulator, to extract, from the script texts of the restrictions on the joint utilities of the multiple software components within each component package, the script portions on the utilized components mentioned in the list (step e), to judge the relationships among the component packages inclusive of said utilized components (step f), to fuse the extracted restriction script texts based on the ascertained relationships among the component packages (step g), and to formulate a restriction script text on the joint uses of the utilized components (component utility rules) (step g).

Figure 7 shows an example of restriction script text formulated by this inter-component utility restriction fusion unit (4) (component utility rules).

The following is expressed by this text:

"The components must be used in the prescribed order, although another component(s) may intervene in-between." /4

Thus, the rule generation device of this application example for utilizing a software component is capable of automatically formulating restriction script texts on multiple components utilized jointly for the formulation of a program, and in a case where such texts are used as test specifications, it becomes possible to automate the program verifying operation.

In a case where the present device is integrated with a component synthesizing system, furthermore, the component synthesis can be executed in a concomitantly verified state, and it becomes possible to formulate a program with a high reliability in a highly efficient manner.

(Effects of the invention)

As the foregoing explanations have demonstrated, the rule generation device of the present invention for utilizing a software component is capable of automatically formulating, by using a formal language, restriction script texts on the joint utilities of multiple software components utilized for the aforementioned program formulation, as a result of which it becomes possible to automate the program verifying operation.

4. Brief explanation of the figures

Figure 1 is a block diagram provided for explaining the constitution of an application example of the rule generation device of the present invention for utilizing a software component. Figure 2 is a flow chart provided for explaining the actions of the rule generation device for utilizing a software component shown in Figure 1, whereas Figure 3 is a diagram which shows the relationships among the respective packages, whereas Figure 4 is a diagram which shows an example of package, whereas Figures 5 and 6 are each diagrams which show examples of restriction script texts on joint utilities of multiple components within a package, whereas Figure 7 is a diagram which shows an example of component utility rules generated by the rule generation device for utilizing a software component shown in Figure 1.

Figure 1.

(1): Component packaging unit; (2): Inter-component utility restriction scripting unit; (3): Inter-package relationship definition unit; (4): Inter-component utility restriction fusion unit.

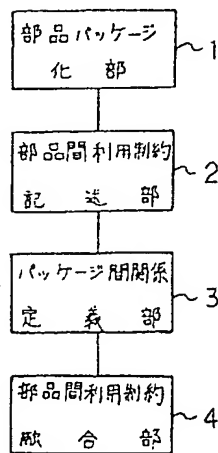
Applicant: Toshiba Corp.

Inventor: Toshio

Agent: Saichi Suyama, patent attorney

Agent: Saichi Suyama

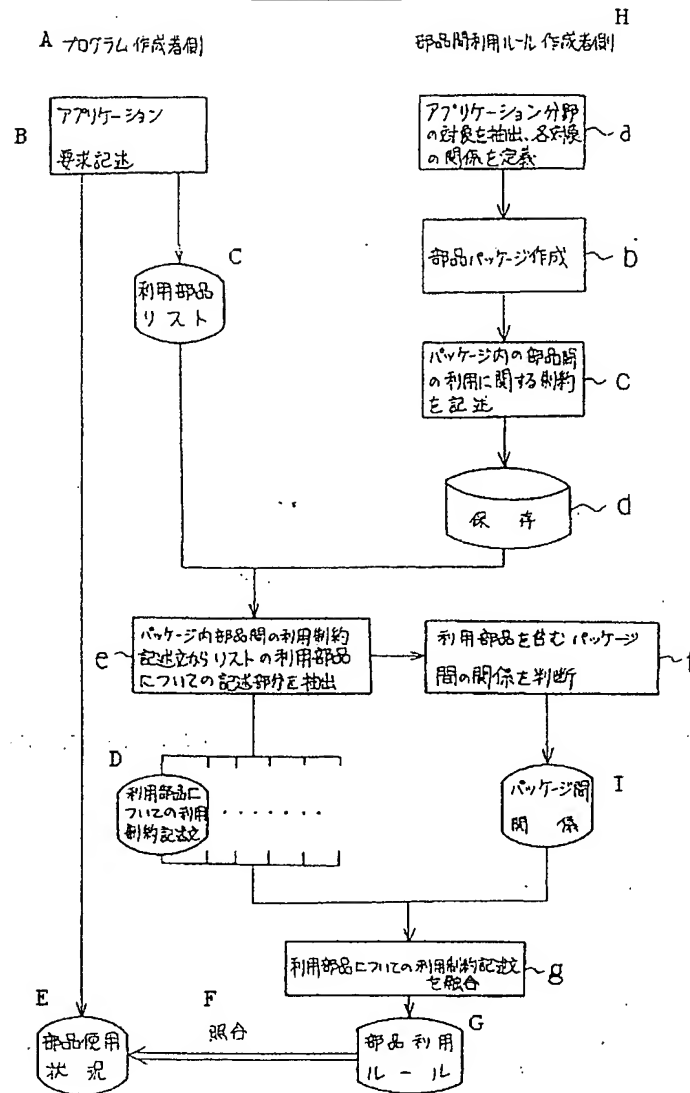
Figure 1



[(1): Component packaging unit; (2): Inter-component utility restriction scripting unit; (3): Inter-package relationship definition unit; (4): Inter-component utility restriction fusion unit]

Figure 2

/5

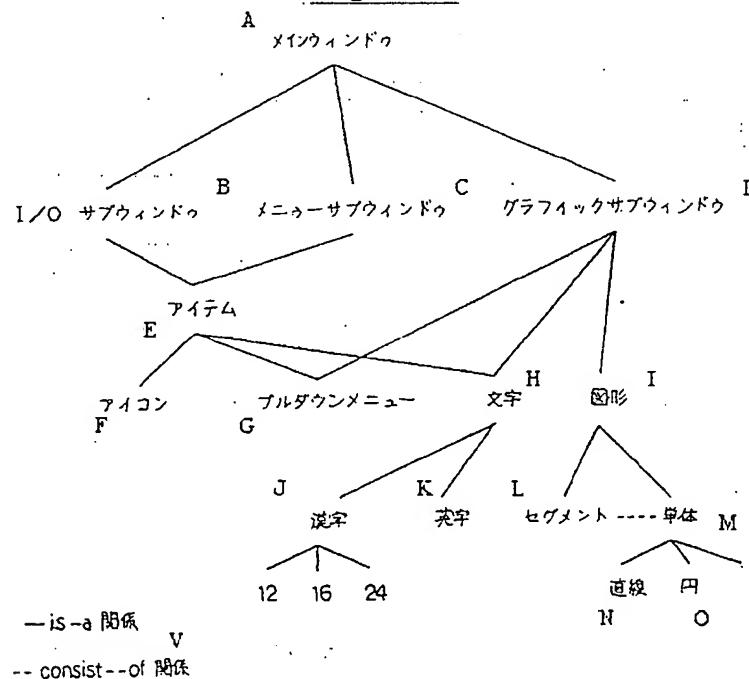


第 2 図

[(A): Program formulator side; (B): Application request script; (C): Utilized component list; (D): Utility restriction script text on utilized components; (E): Component utility state; (F): Matching; (G): Component utility rule; (a): Extraction of targets within an application field & definition of relationships among

the respective targets; (b): Component package formulation; (c): Script on restrictions on joint utilities of multiple components within a package; (d): Storage; (e): Extraction, from the script texts of the restrictions on the joint utilities of multiple software components within each component package, script portions on the utilized components mentioned in the list; (f): Judgment of relationships among packages inclusive of utilized components; (g): Fusion of utility restriction script texts on the utilized components]

Figure 3



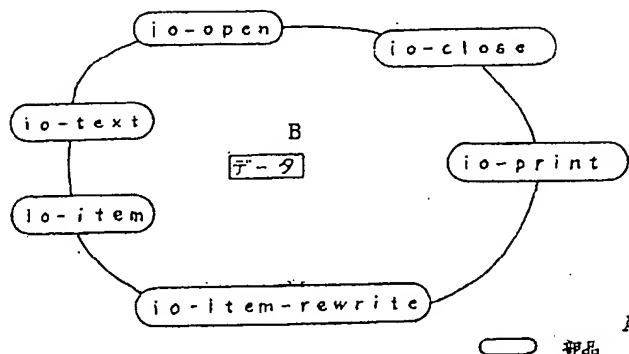
第 3 図

[(A): Main window; (B): I/O sub-window; (C): Menu sub-window; (D): Graphic sub-window; (E): Item; (F): Icon; (G): Pull-down menu;

(H): Characters; (I): Patterns; (J): Kanji characters; (K): European characters; (L): Segment; (M): Singular entity; (N): Line; (O): Circle; (V): Relationship]

Figure 4

/4



第 4 図

[(A): Component; (B): Data]

[(A): Component; (B): Data]

Figure 5

INIT: window-create
END: window-destroy

第 5 図

Figure 6

INIT: io-open
END: io-close

第 6 図

Figure 7

order:window_create,io_open,io_close,window_destroy

第 7 回